Parallel Multi-Physics Integration Frameworks for Virtual Rocket Science

M. Campbell[a], X. Jiao[a], L. Kale[b], O. Lawlor[b], E. de Sturler[b]

[a]Center for the Simulation of Advanced Rockets (CSAR)
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801
{mtcampbe|jiao}@csar.uiuc.edu

[b]Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801
{kale|olawlor|sturler}@cs.uiuc.edu

We are exploring several methodologies, embodied in frameworks, that assist the development of parallel application codes for physical simulations. Those methodologies aim at providing the numerical and parallel software support to integrate independently developed parallel application codes, facilitating easy development of parallel modules, and providing automatic load balancing. We will present an overview of these efforts.

Our integration framework provides a flexible mechanism for inter-module data exchange and function invocation in parallel multi-physics simulations. It maximizes concurrency in development of modules, minimizes code changes for integration, and simplifies interoperability between programming languages (C, C++, and Fortran 90). It allows plug-n-play of alternative physics and computer science modules in an integrated system and rapid prototyping of coupling algorithms. It also provides automatic tracing and profiling capabilities for debugging and performance tuning of complex large-scale simulation codes.

To support coupling algorithms, our framework provides service utilities, such as mapping and transfer of data between disparate meshes and conservative interpolation, numerical operations to manipulate jump conditions, and high-level parallel I/O facilities for restart and visualization. We are also developing numerical components for error estimation and quality control, and for iterative solvers. For the rocket simulation code, our framework has substantially sped up system integration and provided flexibility in rapid prototyping of coupling schemes without sacrificing runtime performance.

Adaptive MPI (AMPI) insulates the application developer from the issue of load balancing. This is one of the most important current issues in parallel simulations, with adaptive and dynamic mesh refinements, multiple time-stepping algorithms, and multi-physics modules complicating load balancing significantly. Programs are written in normal MPI, with a few stylistic requirements, and linked with the AMPI library. By running a large number of MPI "threads", which are extremely light-weight and which can be migrated across processors, the runtime system can automate checkpointing, out-of-core execution, changing the number of processors used at runtime, in addition to dynamic load balancing.

Fueled by the observation that a large class of parallel simulation codes are based on a small group of parallel data-structures and operations, we have developed several frameworks. Each framework captures essential features needed for a class of applications (such as those based on unstructured meshes), and avoids the need to repeatedly code it in individual applications. We currently have frameworks for structured meshes, and unstructured meshes, while a particle framework, and AMR support is under development.

AMPI and the component frameworks developed at the Parallel Programming Laboratory are also used for several applications outside the CSAR. More information can be found at http://www.csar.uiuc.edu and http://charm.cs.uiuc.edu.